Using Bindings to Help Sync Your Data and UI

Session 113

These are confidential sessions—please refrain from filming or taking pictures.

Application Technologies

Using Bindings to Help Sync Your Data and UI

Malcolm Crawford
Technical Writer

http://developer.apple.com/wwdc2005/

Introduction

Goal: to answer the following questions

- What are Cocoa bindings?
- What problem do they solve?
- How do they work?
- What are controllers and value transformers?

What You'll Learn

- How to establish bindings in Interface Builder and in code
- What are the following technologies and why are they useful
  - Key-value binding
  - Key-value coding
  - Key-value observing
- The principles behind implementing a custom bindings-enabled view
- A little about the rôle of controllers
- How to implement and use value transformers

Answering the First Question

http://developer.apple.com/wwdc2005/

What Are Cocoa Bindings?

- A set of technologies for keeping Models and Views synchronized

What Problems Do Bindings Solve?

http://developer.apple.com/wwdc2005/

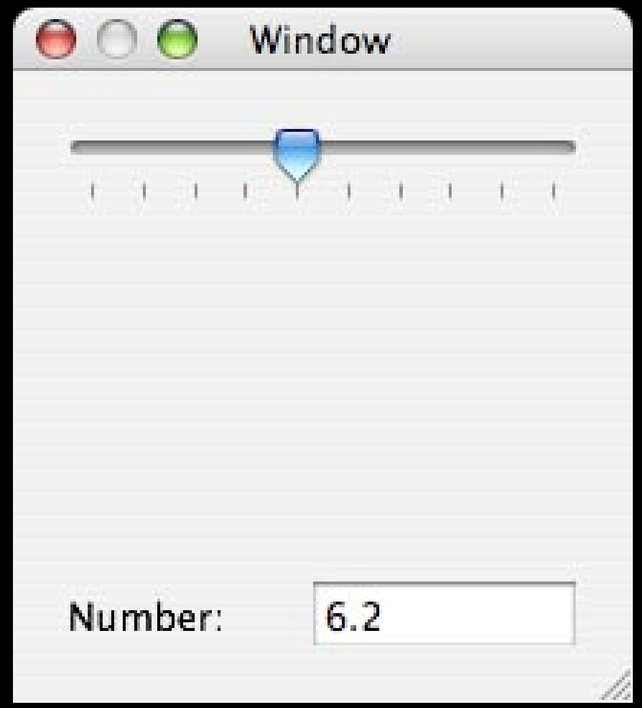
Glue Code

Much code deals with keeping user interface and data in sync

- Many outlets, delegates
- Routine work
- Repetitive

DataObject

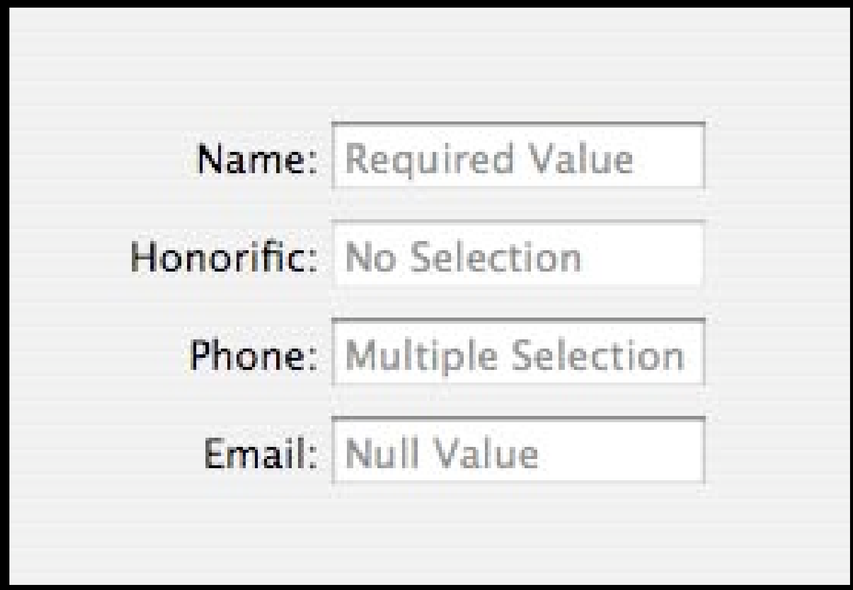
number = 6.2



User Interface Friendliness

Common user interface states handled cleanly

- Informative user interfaces
- Marker values for the following states:
  - Null value (nil)
  - Multiple values selection
  - No selection
  - Not applicable
- Define placeholder values for each state as needed/desired

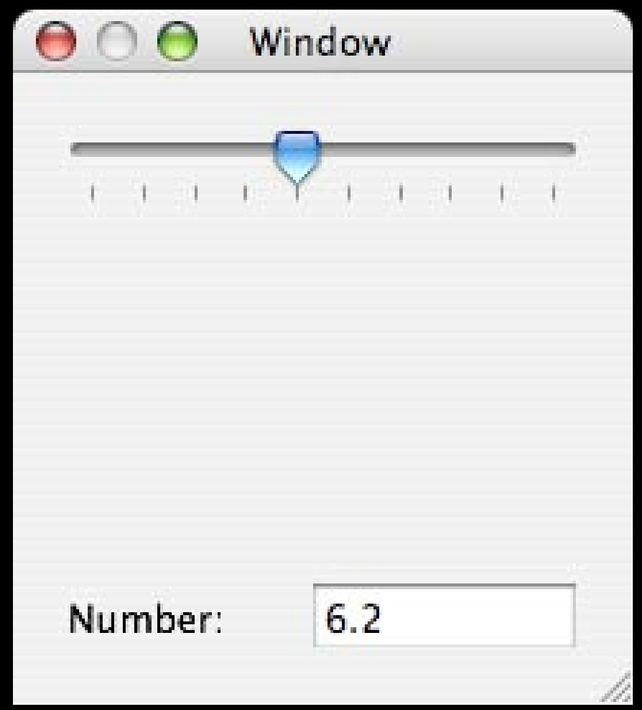


Demo Goal

- Keep user interface and data in sync
  - Direct target-action
  - Traditional Cocoa
  - Bindings

NSMutableDictionary

number = 6.2



Demo

http://developer.apple.com/wwdc2005/

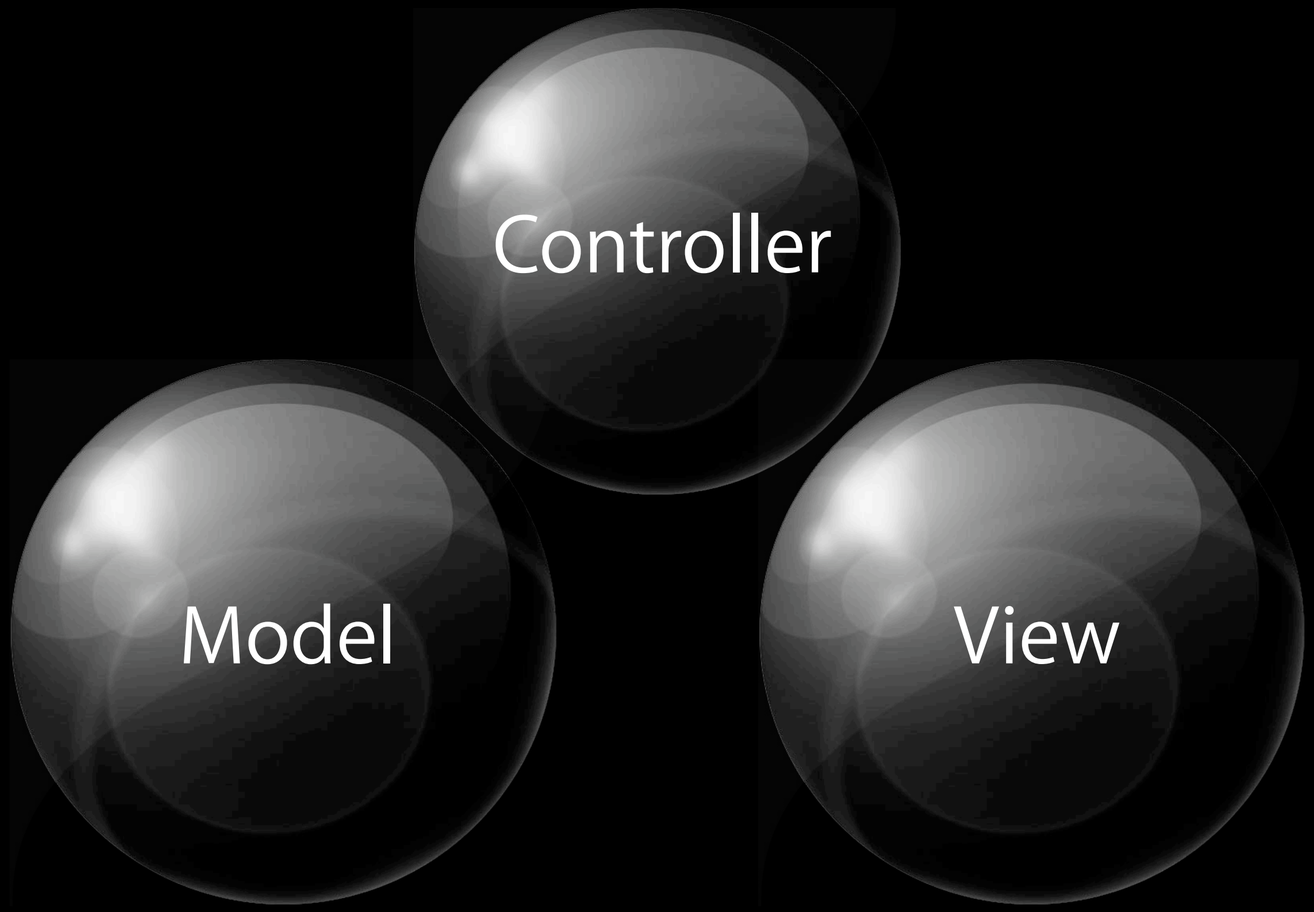
Model, View, Controller

Recap on the demo

http://developer.apple.com/wwdc2005/

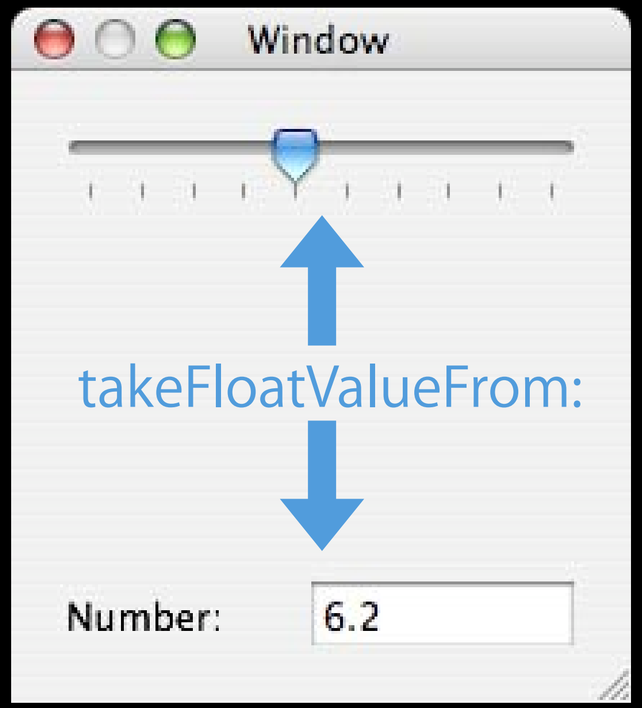
Model, View, Controller

A design pattern to factor code



Not an Example

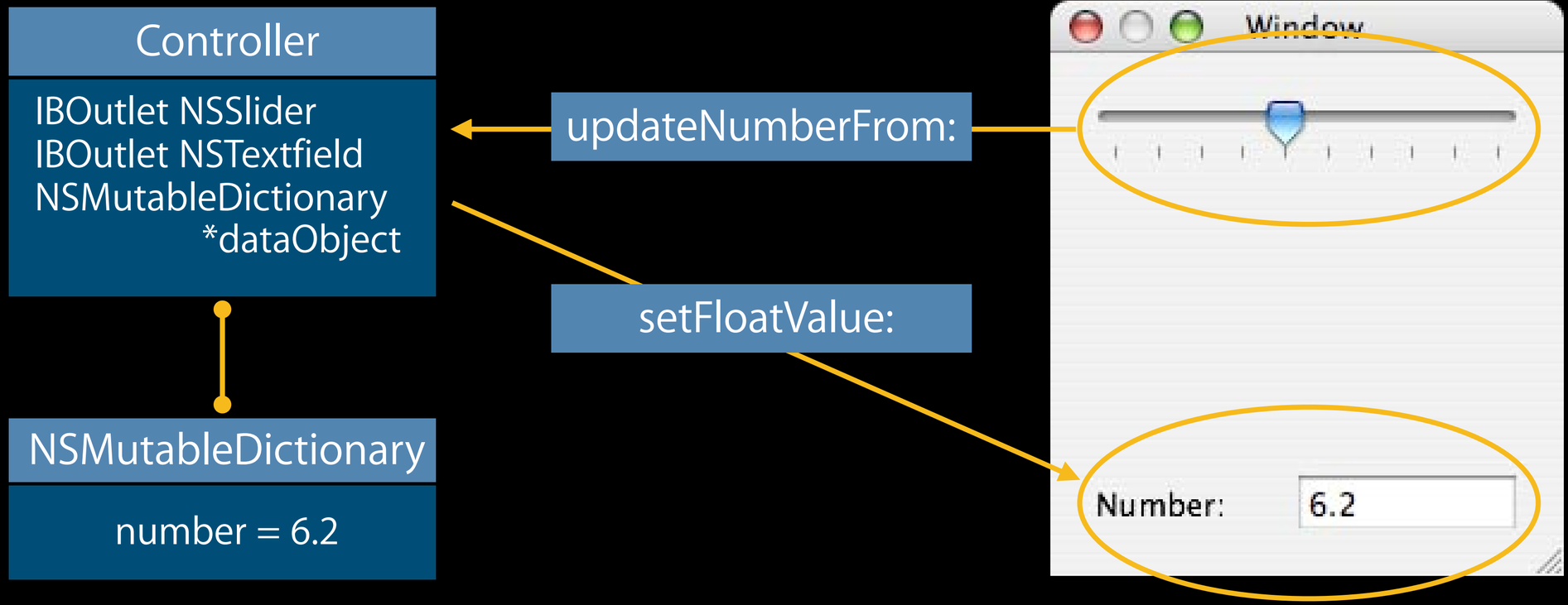
- Connect slider to text field
  - Values kept in sync
- Use target-action
- Fatally flawed



Traditional Cocoa

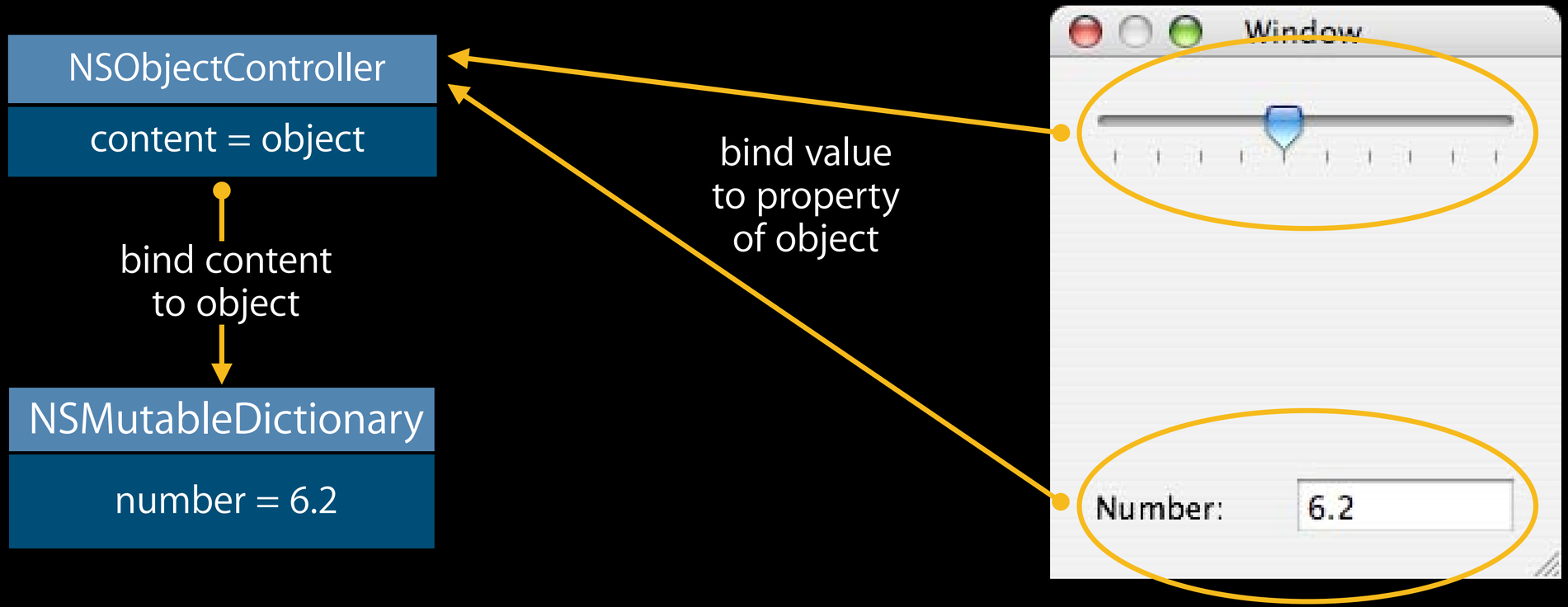
Adheres to MVC pattern

- Target-action
- Outlets
- Glue



Cocoa Bindings

- NS*Controller
- No outlets
- Not target-action
- No glue



Cocoa and MVC

- Models are reusable
- Views are resuable
- Now controllers are reusable
  - NSObjectController
  - NSArrayController
  - NSUserDefaultsController

When Do You Need To Do Any Work?(!)

- Implementing custom model classes
- Implementing custom views
- Implementing controller subclasses
- Establishing bindings programatically
- Implementing and initializing value transformers
- Factoring your code…

What Is a Binding?

What Is a Binding?

- A collection of information that specifies what attribute of one object should be kept synchronized what property of another with what options
- Bindings are “established,” not “created”
  - Think target-action…

Binding Types

- Single-value bindings
- Multiple-value bindings
  - Construct final value by combining values of multiple properties

Multiple-Value Bindings

- Establishing the first binding automatically causes second binding to be exposed, and so on
- Always read-only
- Three variations
  - Value with pattern bindings
    - Assemble final value from multiple string values
    - Common use: indicate selection in the form “5 of 34 selected”
  - Multiple-value Boolean bindings
    - Assemble final value from multiple Boolean values
    - Common use: enable this button if this and that are true
  - Multiple-value argument bindings
    - Replacement for target-action

Options

- Value transformers
  - Later…
- Selection markers
  - Allow you to specify values that apply if:
    - No selection
    - Null value
    - Multiple selection…
- Additional parameters
  - Read the documentation

How Do Bindings Work?

“Progressive JPEG Intro”

http://developer.apple.com/wwdc2005/

Overview

- Establishing bindings
- What are the underlying technologies?
- Value updates
- Decomposing a binding
- What do you need to do to support bindings?

Design Principles

- Requirement: must work with any class
- Relies on three underlying technologies
  - Key-value coding
  - Key-value observing
  - Key-value binding

How Do You Establish Bindings?

- In Interface Builder
- In code

} Key-value binding protocol

Key-Value Binding Protocol

- Establish a binding

```
- (void)bind:  (NSString *) toObject:  (id)
withKeyPath:  (NSString *)
options:  (NSDictionary *)
```

- Remove a binding

```
- unbind:
```

  Rare

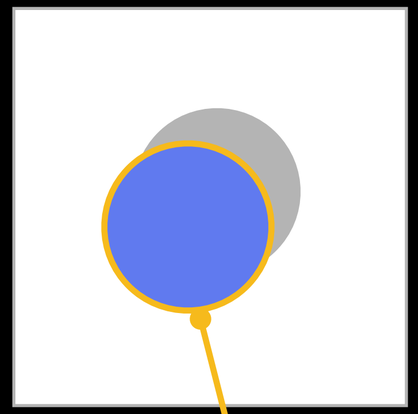
- Expose bindings

```
+ exposeBinding:, - exposedBindings
```

- Class validation

```
- valueClassForBinding:
```

Joystick Example



NSController

content = graphic

content bound to graphic

Graphic

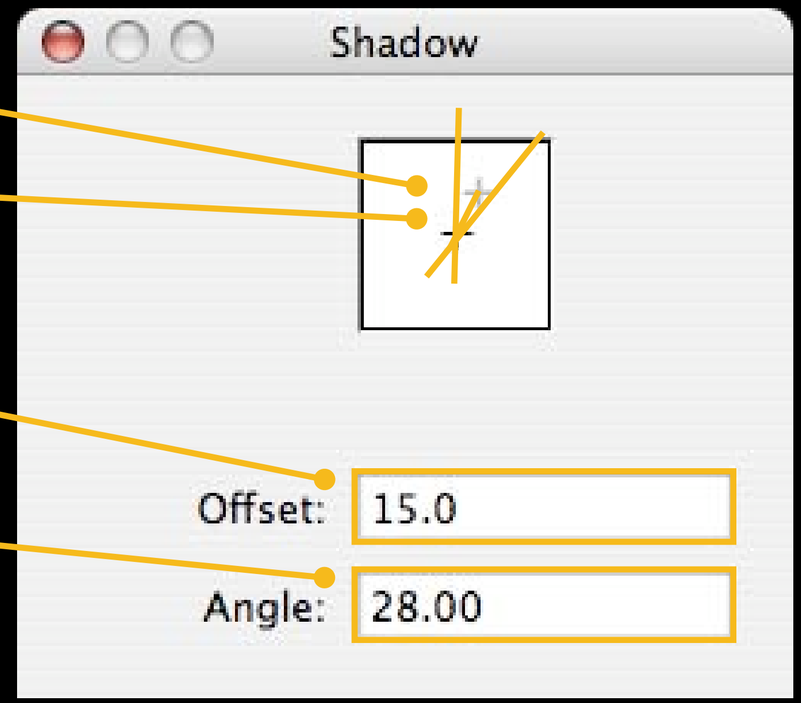
shadowOffset = 15.00
shadowAngle = 0.49

angle bound to selection.shadowOffset with degrees to radians value transformer

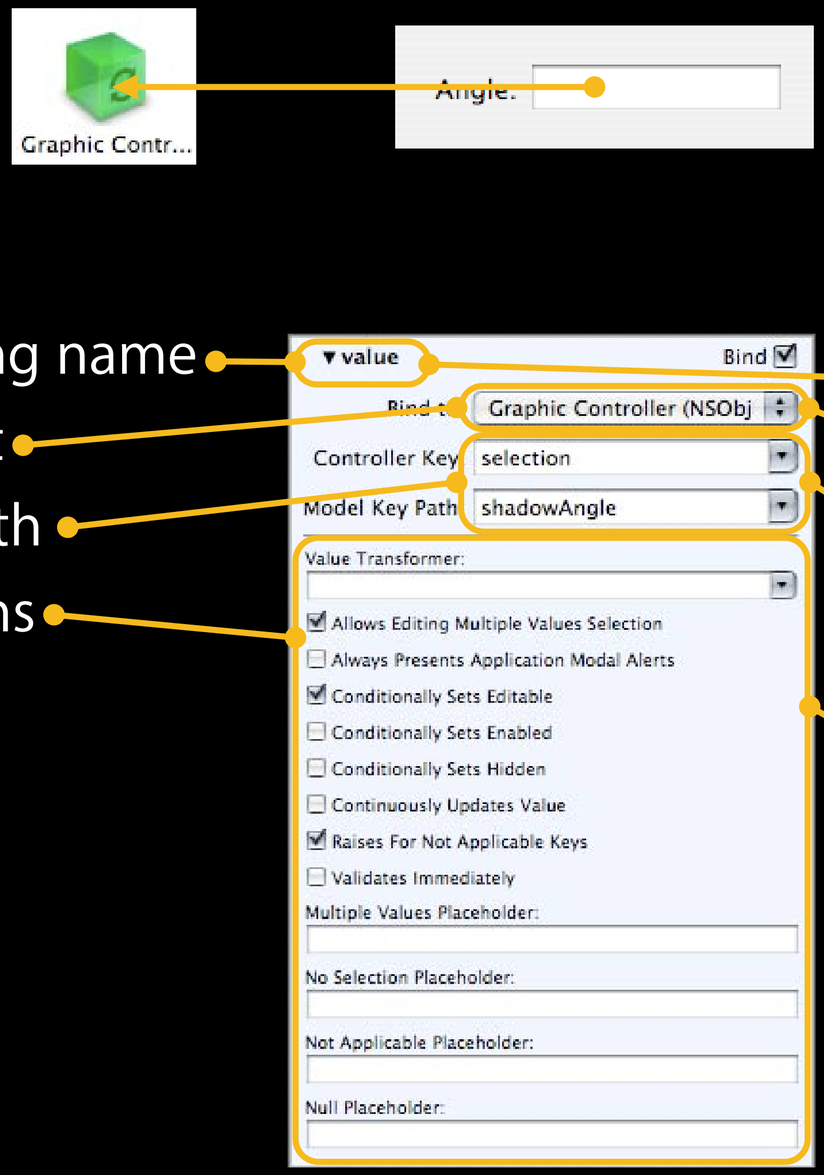
offset bound to selection.shadowOffset

value bound to selection.shadowOffset

value bound to selection.shadowOffset with degrees to radians value transformer



Establishing a Binding



- Bind text field's value to graphic controller with keypath selection.shadowAngle with options…

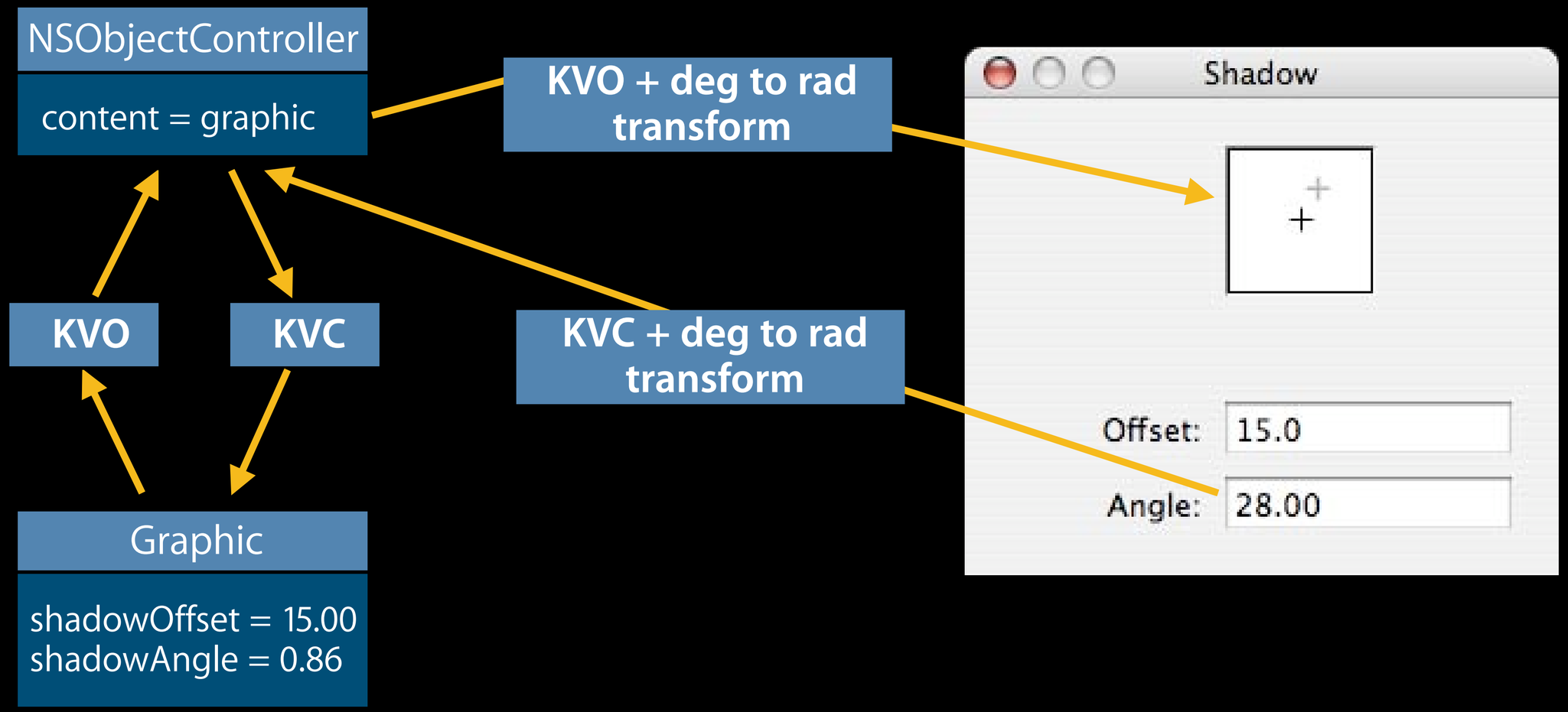
```
[textfield bind: @"value"
      toObject: graphicController
   withKeyPath: @"selection.shadowAngle"
       options: optionsDictionary];
```

Other Supporting Technologies

- Key-value coding
- Key-value observing
- Consider a complete edit cycle…

The Complete Edit Cycle

Textfield to model



Edit Cycle Summary

- Key-value coding
  - Pushes values from UI to model
- Key-value observing
  - Announces changes of model property values to UI

Key-Value Coding

- Unified means of accessing objects' properties by key
  - `-valueForKey:`, `-setValue:forKey:`
  - Key is a string
  - No need to know custom accessors
  - Flexible runtime binding
- Call appropriate accessor if it exists
  - `valueForKey:@"firstName"` -> `firstName`
  - `setValue:newName forKey:@"firstName"` -> `setFirstName:newName`

- Collections have special semantics…

To-Many Relationships via KVC

- `- (NSMutableArray *)mutableArrayValueForKey:(NSString *)key`
- `- (NSMutableSet *)mutableSetValueForKey:(NSString *)key`

- Return a proxy mutable collection
  - Modify the proxy

```
NSMutableArray *shapes =
        [canvas mutableArrayValueForKey:@"shapes"];
[shapes addObject:shape];
```

Key Paths

- Key paths allow you to follow relationships
  - `valueForKeyPath:, setValue:forKeyPath:`
- Employee has firstName and manager
  - `[employee valueForKey: @"firstName"]`
  - `[employee valueForKeyPath: @"manager.firstName"]`

KVC Validation

- `- (BOOL)validateValue:(id *)ioValue forKey:(NSString *)key error:(NSError **)outError`
- Implement property-specific methods in your model class
  - `validate<Key>:(id *)ioValue error:(NSError **)outError`
- If value OK, return YES
- Otherwise return NO and specify an error
- Can modify the proposed value and return YES
  - Not recommended

KVC to Send Updates

V→C→M

Update to value in controller passed to model using KVC

NSController

content = graphic

setValue: forKeyPath:

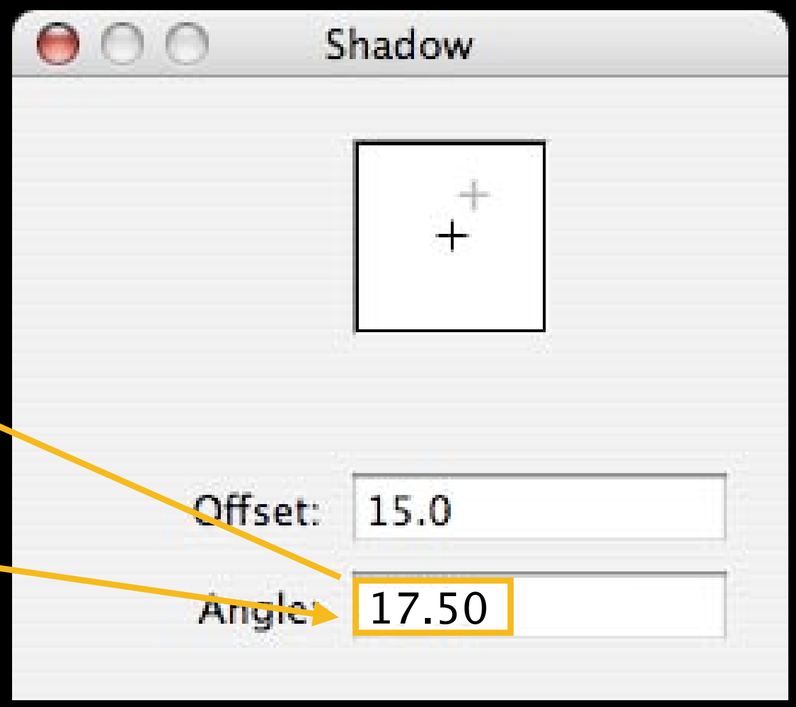
Graphic

shadowOffset = 15.00
shadowAngle = 0.49

setValue: forKeyPath:

Validation

Update to value in view passed to controller using KVC



KVC: What Do You Need to Do?

- Must ensure model classes are KVC-compliant
  - Also any custom controller classes
- Implement suitable accessor methods
  - Notably for arrays

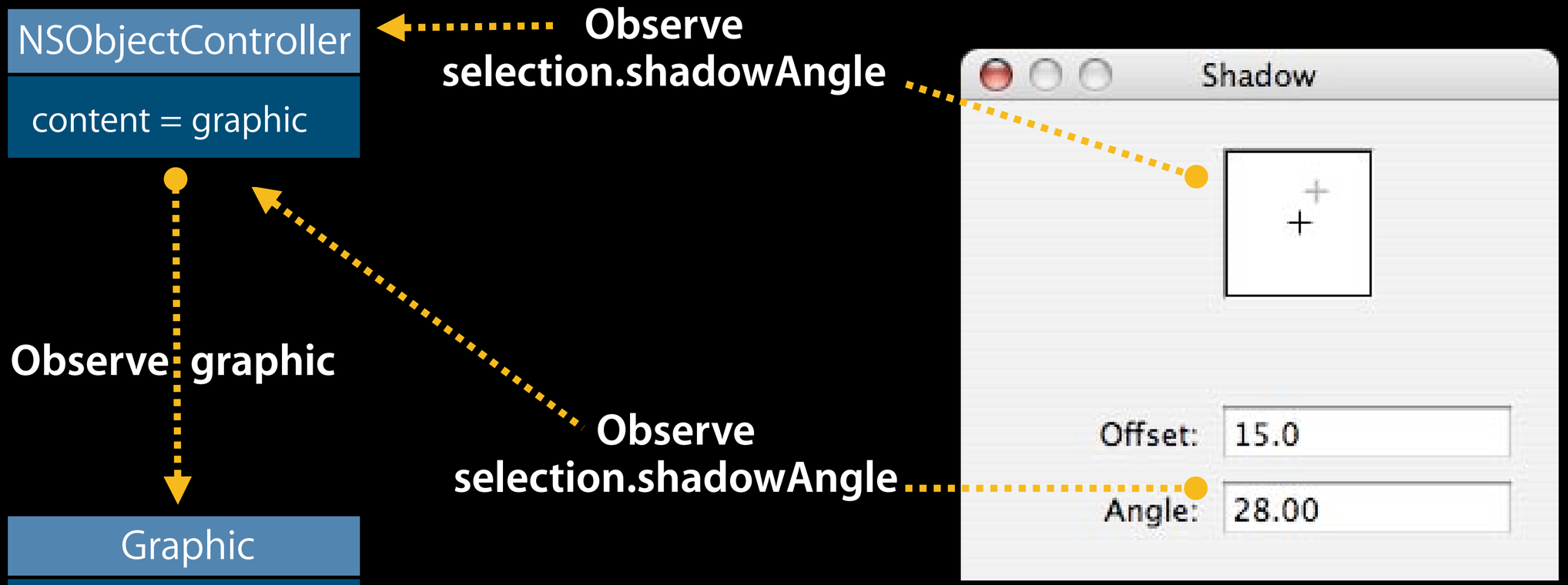
Key-Value Observing

- Mechanism that allows an object to receive notifications of changes to values in other objects
- Two parts
  - Registration
    - `addObserver:forKeyPath:options:context:`
  - Notification
    - `observeValueForKeyPath:ofObject:change:context:`
  - Direct message—not NSNotificationCenter

KVO—Registration

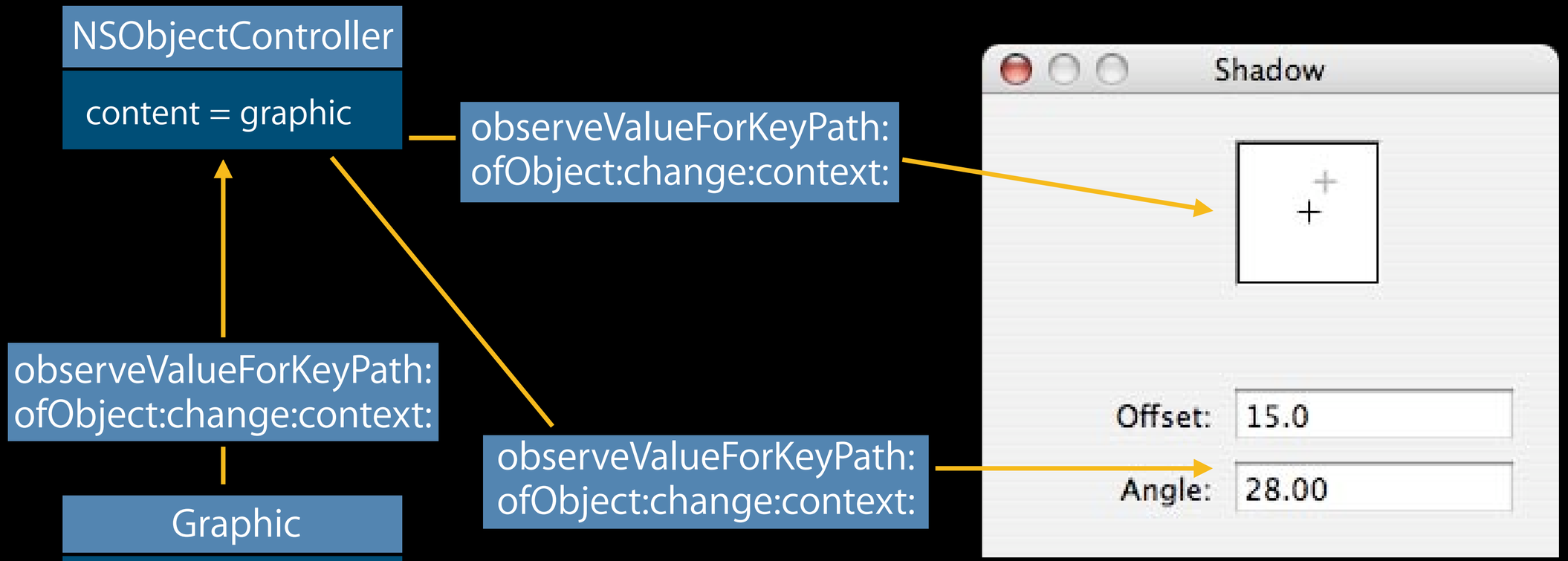
Register V→C→M

`addObserver:forKeyPath:options:context:`



KVO—Observation

Change Notification M→C→V



KVO: What Do You Need to Do?

- Support added automatically by runtime
  - Implement suitable accessor methods
  - Same as for KVC compliance
- Can also implement manual support
  - NSKeyValueObserving:
  - `- willChangeValueForKey:`
  - `- didChangeValueForKey:`

Set Accessor With Manual KVO

```
- (void)setName:(NSString *)newName
{
  if (name != newName)
  {
    [self willChangeValueForKey:@"name"];
    [name release];
    name = [newName copy];
    [self didChangeValueForKey:@"name"];
  }
}
```

Technologies Summary

- Supporting technologies: KVC, KVO, KVB
- Use KVB directly to create binding
- KVC support through following accessor method naming conventions
- KVO support added automatically

In More Detail

http://developer.apple.com/wwdc2005/

More Details

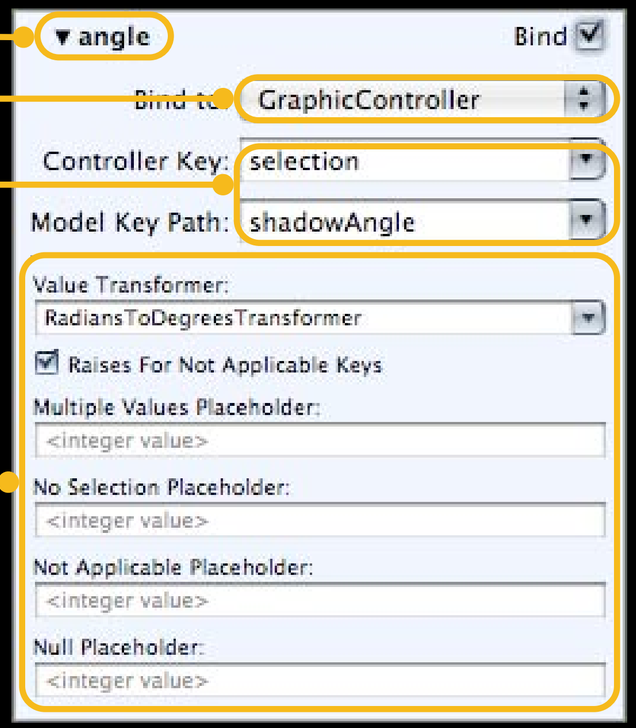
- Examine infrastructure in greater detail
- Understand in principle how you implement a bindings-enabled view
  - Consider “joystick” view example

Establishing a Binding

- What does view have to do when an attribute is bound?
- Recall target-action analogy
  - Must record information

Angle Binding

bindingName
object
keyPath
options




```
[joystick bind:  @"angle"
     toObject:
          graphicController
     withKeyPath:
          @"selection.shadowAngle"
     options:
          optionsDictionary];
```

- Need to record this information

Joystick Class Declaration

```
@interface Joystick : NSView
{
  NSNumber *angle;
  id observedObjectForAngle;
  NSString *observedKeyPathForAngle;
  NSValueTransformer *angleValueTransformer;
  // ...
}

static void *AngleObservationContext = (void *)@"JoystickAngle";
```

Joystick's bind:… Implementation

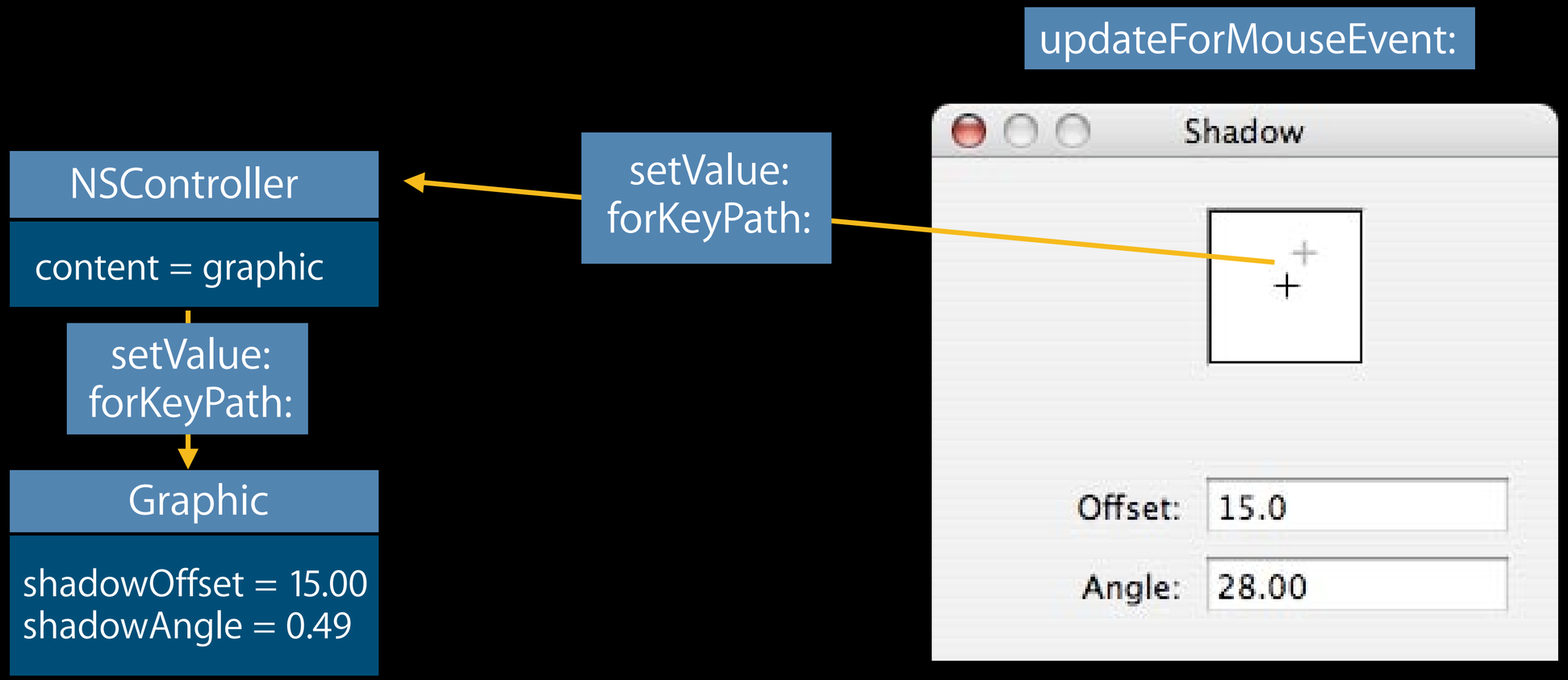
```
- (void)bind:(NSString *)binding  toObject:(id)observableController
    withKeyPath:(NSString *)keyPath  options:(NSDictionary *)options
{
    if ([binding isEqualToString:@"angle"]) {

        [observableController addObserver: self    forKeyPath: keyPath
          options: nil    context: AngleBindingIdentifier];

        [self setObservedObjectForAngle: observableController];
        [self setObservedKeyPathForAngle: keyPath];
        NSString *vtName = [options objectForKey:@"NSValueTransformerName"];
        angleValueTransformer = [NSValueTransformer valueTransformerForName: vtName];
    }
    // ...
```

Two Perspectives on Updates

- View-initiated
- Model-initiated

View-Initiated Update

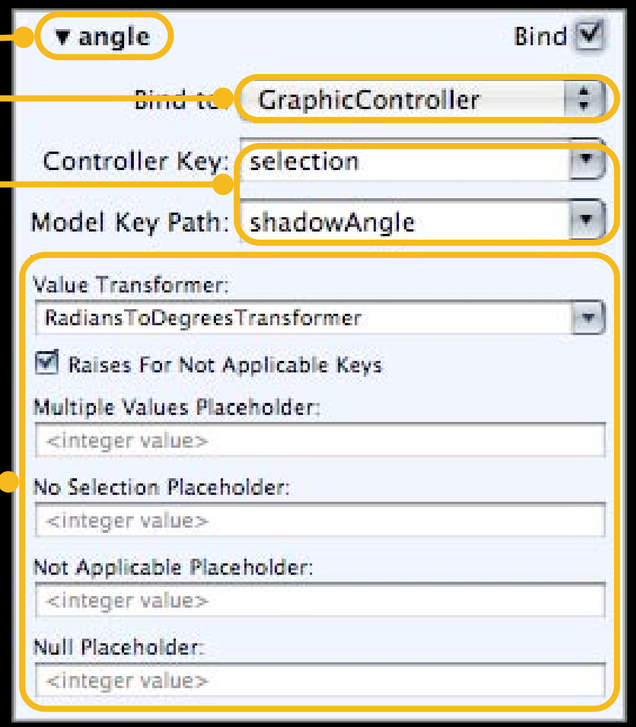
V→C→M



Joystick's Binding

View-initiated update

bindingName
object
keyPath
options




```
[joystick bind:  @"angle"
     toObject:
          graphicController
     withKeyPath:
          @"selection.shadowAngle"
     options:
          optionsDictionary];
```

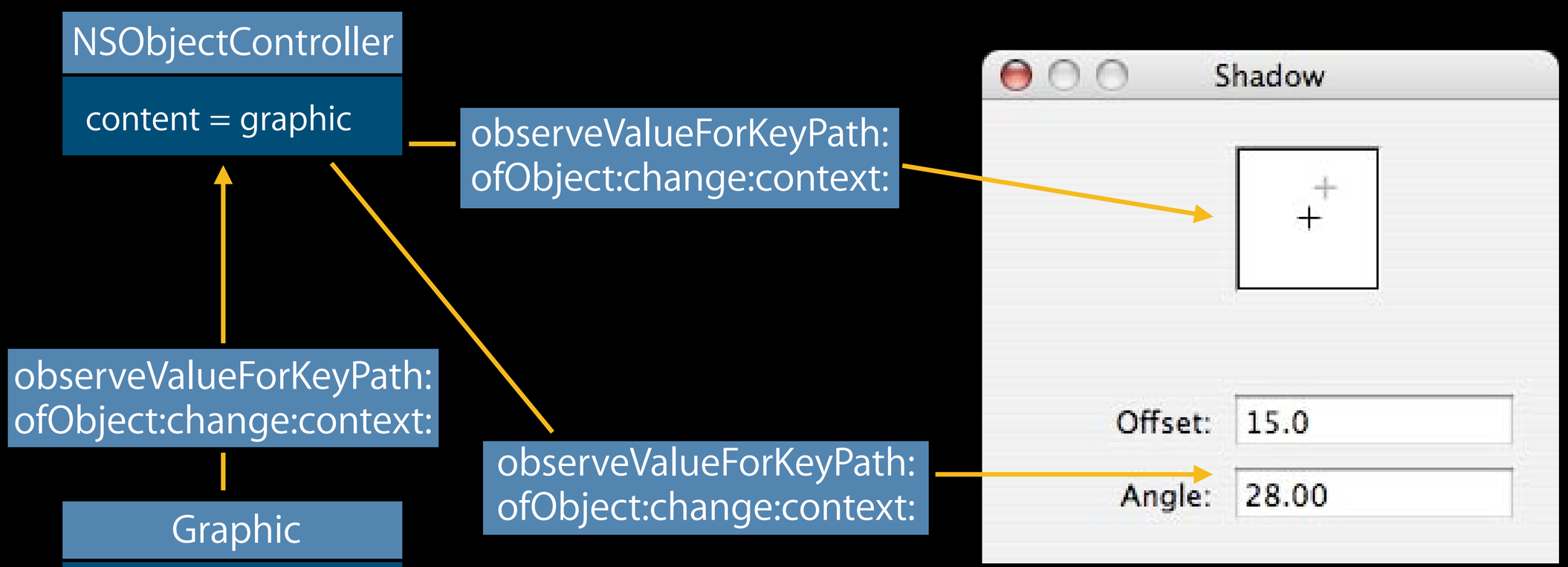
View-Initiated Update

- `bind:`
  - If the value of this attribute changes…
- `toObject:`
  - …tell this object…
- `withKeyPath:`
  - …that this property changed…
- `options:`
  - …using these options

```
-(void)updateForMouseEvent:(NSEvent *)event
{
  NSNumber *newAngle = ... // calculate angle in degrees...
  [self setAngle:newAngle];

  if (observedObjectForAngle != nil) {

    if (angleValueTransformer != nil) {
        newAngle = [vt reverseTransformedValue: newAngle];
    }
    [observedObjectForAngle setValue: newAngle
                          forKeyPath: observedKeyPathForAngle];
  }
  // calculate offset, redisplay...
```

Model-Initiated Update

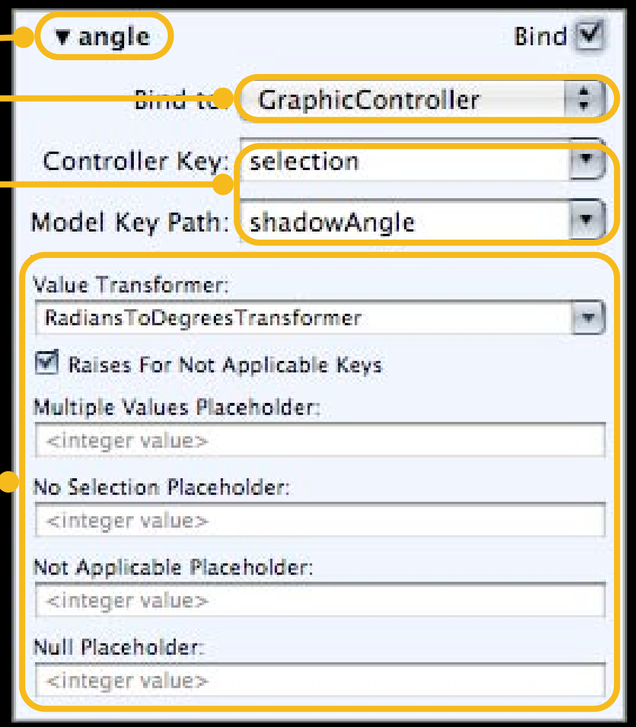
M→C→V



Joystick's Binding

Model-initiated update

bindingName
object
keyPath
options




```
[joystick bind:  @"angle"
     toObject:
          graphicController
     withKeyPath:
          @"selection.shadowAngle"
     options:
          optionsDictionary];
```

Model-Initiated Update

- `withKeyPath:`
  - If this property…
- `toObject:`
  - …of this object changes…
- `bind:`
  - …update this attribute…
- `options:`
  - …using these options

```
- (void)observeValueForKeyPath:(NSString *)keyPath  ofObject:(id)object
        change:(NSDictionary *)change  context:(void *)context
{
  if (context == AngleObservationContext)
  {
     id nA = [observedObjectForAngle valueForKeyPath:observedKeyPathForAngle];
     if (! NSIsControllerMarker (nA)) {

        if (angleValueTransformer != nil) {
           nA = [angleValueTransformer transformedValue:nA];
        }
        [self setValue:nA forKey:@"angle"];
     }
     // else ... [self setNeedsDisplay:YES];
```

Technologies Summary

- Supporting technologies: KVC, KVO, KVB
- Use KVB directly to create binding
- KVC support through following accessor method naming conventions
- KVO support added automatically

Value Transformers

http://developer.apple.com/wwdc2005/

Value Transformers

- Subclasses of NSValueTransformer
  - Lightweight NSFormatter
- Register by name
- System provides basic set
- You can create and register your own…

Using Transformers

- Rather like formatters
  - Simpler to implement
  - `+ (Class)transformedValueClass`
  - `+ (BOOL)supportsReverseTransformation`
  - `- (id)transformedValue:(id)aBool`
  - Reverse transform optional
  - `- (id)reverseTransformedValue:(id)value`
- To use in IB, must register instance
  - `[NSValueTransformer setValueTransformer:transformer forName:@"OverdueTransformer"];`
  - Typically in `initialize`

```
@implementation RadiansToDegreesTransformer

+ (Class)transformedValueClass    {    return [NSNumber self];    }

+ (BOOL)allowsReverseTransformation    {    return YES;    }

- (id)transformedValue:(id)aNumber {
    double radians = [aNumber doubleValue];
    return [NSNumber numberWithDouble: (radians / (3.1415927/180.0))];
}

- (id)reverseTransformedValue:(id)aNumber {
    double degrees = [aNumber doubleValue];
    return [NSNumber numberWithDouble: (degrees * (3.1415927/180.0))];
}
@end
```

Controllers

http://developer.apple.com/wwdc2005/

Controllers

- Object controller
- Array controller
- Preferences controller

What Do Controllers Do?

- Implement placeholders
- Implement NSEditor, NSEditorRegistration
  - Ensure edits are committed before window is closed
- Support selection
  - Provide selection markers…
- Content is dynamic

Selection Markers

- Value passed to observers when bound value is null, invalid, multiple
  - NSNoSelectionMarker
  - NSNotApplicableMarker
  - NSMultipleValuesMarker

Why Subclass?

- Filtering
- Sorting
- Custom initialization
- …

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Related Sessions

Getting the Most Out of Cocoa Bindings	Session 136 Thursday 2:00PM

Who to Contact

Matthew Formica
Cocoa Evangelist
mformica@apple.com

Cocoa Developer List
cocoa-dev@lists.apple.com

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Q&A

Matthew Formica
Cocoa Evangelist
mformica@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference